Docket No.: P-0316 #2 PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of :

Sang Wook KIM and Young Wook JUN :

New U.S. Patent Application :

Filed: December 27, 2001 :

For: A METHOD FOR PROCESSING DYNAMIC DATABASE IN DISTRIBUTED PROCESSING SYSTEM BASED ON CORBA PLATFORM

TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT

Assistant Commissioner of Patents
Washington, D. C. 20231

Sir:

At the time the above application was filed, priority was claimed based on the following application:

Korean Patent Application No. 84627/2000, filed December 28, 2000.

A copy of each priority application listed above is enclosed.

Respectfully submitted,
FLESHNER & KIM, LLP

Daniel Y.J. Kim
Registration No. 36,186
David W. Ward
Registration No. 45,198

P. O. Box 221200
Chantilly, Virginia 20153-1200
703 502-9440
Date: December 27, 2001
DYK/DWW:tmd

대 한 민 국 특 허 청
KOREAN INTELLECTUAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

출 원 번 호 : 특허출원 2000년 제 84627 호
Application Number PATENT-2000-0084627

출 원 년 월 일 : 2000년 12월 28일
Date of Application DEC 28, 2000

CERTIFIED COPY OF PRIORITY DOCUMENT

출 원 인 : 엘지전자주식회사
Applicant(s) LG ELECTRONICS INC.

2001 년 12 월 07 일

특 허 청
COMMISSIONER

【서지사항】

【서류명】	특허출원서
【권리구분】	특허
【수신처】	특허청장
【참조번호】	0038
【제출일자】	2000.12.28
【발명의 명칭】	코바 플랫폼을 이용한 동적 데이터베이스 처리방법
【발명의 영문명칭】	Method for dynamic database processing by using CORBA platform
【출원인】	
【명칭】	엘지전자 주식회사
【출원인코드】	1-1998-000275-8
【대리인】	
【성명】	홍성철
【대리인코드】	9-1998-000611-7
【포괄위임등록번호】	2000-049936-1
【발명자】	
【성명의 국문표기】	김상욱
【성명의 영문표기】	KIM,SANG WOOK
【주민등록번호】	710809-1829817
【우편번호】	412-722
【주소】	경기도 고양시 덕양구 행신1동 햇빛마을 2406동 1203호
【국적】	KR
【발명자】	
【성명의 국문표기】	전영욱
【성명의 영문표기】	JEON,YOUNG WOOK
【주민등록번호】	710830-1023810
【우편번호】	110-523
【주소】	서울특별시 종로구 명륜동3가 85-13 101호
【국적】	KR
【취지】	특허법 제42조의 규정에 의하여 위와 같이 출원합니다. 대리인 홍성철 (인)

【수수료】

【기본출원료】	19	면	29,000 원
【가산출원료】	0	면	0 원
【우선권주장료】	0	건	0 원
【심사청구료】	0	항	0 원
【합계】	29,000 원		

【요약서】

【요약】

본 발명은 CORBA 플랫폼을 이용한 동적 데이터베이스 처리방법을 제공하기 위한 것으로, 클라이언트에서는 자신이 데이터베이스에 접근하기 위해 SQL 스트링을 구성한 다음 구성한 스트링과 참조 포인터를 파라미터로 서버의 범용 매소드를 호출하는 제 1 단계와; 상기 서버의 매소드는 SQL 스트링을 이용하여 데이터를 질의하여 데이터베이스로부터 받아온 하나의 row 데이터를 로컬 메모리에 링크드 리스트 형태로 저장하고, 한의 row에 대한 선택 작업 완료 후 범용 데이터 형의 클래스를 인스턴스 생성하고, 어레이로 메모리를 생성시켜 링크드 리스트에 있는 데이터를 할당하는 제 2 단계와; 상기 제 2 단계에서의 질의 작업이 완료되면 상기 클라이언트는 참조 포인터를 이용해 상기 서버가 할당한 결과값에 접근하는 제 3 단계를 포함하여 구성함으로써, CORBA를 이용한 분산 처리 시스템에서 서버와 클라이언트 간에 데이터 처리에서 데이터베이스 검색을 위한 범용 IDL을 정의하고 서버에서의 처리를 동적으로 처리할 수 있게 함으로써 개발자의 개발 범위를 최소화하고 이후에 변경되는 클라이언트의 트랜잭션 요구에 대해서 IDL의 변경을 최소화할 수 있게 되는 것이다.

【대표도】

도 4

【명세서】

【발명의 명칭】

코바 플랫폼을 이용한 동적 데이터베이스 처리방법{Method for dynamic database processing by using CORBA platform}

【도면의 간단한 설명】

도 1은 종래 CORBA 플랫폼을 이용한 데이터베이스 처리를 보인 블록구성도이고,

도 2는 도 1에서 각 매소드의 IDL 예를 보인 도면이며,

도 3은 종래 CORBA 플랫폼을 이용한 데이터베이스 처리방법을 보인 흐름도이고,

도 4는 본 발명에 의한 CORBA 플랫폼을 이용한 동적 데이터베이스 처리방법을 보인 흐름도 이며,

도 5는 도 4에 의한 CORBA 플랫폼에서의 동적 처리 구조를 보인 도면이고,

도 6은 도 4에 의한 동적 서버의 동작 구조를 보인 도면이며,

도 7은 도 4에 의한 범용 매소드의 IDL과 리턴값의 구조를 보인 도면이다.

* 도면의 주요 부분에 대한 부호의 설명 *

10 : 클라이언트

20 : 서버

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<11> 본 발명은 CORBA(Common Object Request Broker Architecture) 플랫폼을 이용한 동적 데이터베이스 처리방법에 관한 것으로, 특히 CORBA를 이용한 분산 처리 시스템에서 서버(Server)와 클라이언트(Client) 간에 데이터 처리에서 데이터베이스 검색을 위한 범용 IDL(Interface Definition Language)을 정의하고 서버에서의 처리를 동적으로 처리할 수 있게 함으로써 개발자의 개발 범위를 최소화하고 이후에 변경되는 클라이언트의 트랜잭션(Transaction) 요구에 대해서 IDL의 변경을 최소화하기에 적당하도록 한 CORBA 플랫폼을 이용한 동적 데이터베이스 처리방법에 관한 것이다.

<12> 일반적으로 CORBA 플랫폼(platform) 하에서는 서버(Server)와 클라이언트(Client)가 데이터베이스를 처리하는 서버의 오브젝트(Object)의 참조 포인터를 구한 후 그 포인터를 통해 서버(Server)의 매소드(Method)를 호출한다.

<13> 호출된 매소드(Method)는 요구된 SQL(Structured Query Language)을 처리하고, 그 결과값을 클라이언트에 전달하기 위해 IDL(Interface Definition Language)에 정의되어 있는 out 데이터 형을 시퀀스 타입(Sequence type)으로 구성하게 된다.

<14> 이러한 out 데이터는 클라이언트(Client)가 필요로 하는 매소드(Method) 마다 정의를 해야한다.

<15> 그리고 종래 CORBA 플랫폼을 이용한 데이터베이스 처리와 각 매소드의 IDL 예는 도 1 및 도 2에 도시하였다.

<16> 여기서 참조번호 10은 클라이언트이고, 20은 서버이다.

<17> 도 3은 종래 CORBA 플랫폼을 이용한 데이터베이스 처리방법을 보인 흐름도이다.

<18> 이에 도시된 바와 같이, 클라이언트(10)가 서버(20)의 오브젝트의 참조 포인터를 구한 다음 상기 클라이언트(10)는 참조 포인터를 통해 상기 서버(20)의 매소드를 호출하는 단계(ST11)(ST12)와; 상기 서버(20)는 클라이언트(10)로부터 SQL 데이터를 전달받아 해당 매소드를 실행하여 데이터베이스로부터 데이터를 추출하는 단계(ST13)(ST14)와; 상기 추출된 데이터의 데이터 형을 파라미터로 전달받은 구조체 레퍼런스의 시퀀스 형으로 설정하고, 메모리를 할당하며, 데이터를 복사하는 단계(ST15 ~ ST17)와; 상기 클라이언트(10)는 파라미터를 전달한 구조체 레퍼런스를 통해 반환값을 가져와 처리하는 단계(ST18)를 수행한다.

<19> 그래서 종래의 데이터베이스 접근 방법은 먼저 클라이언트(10)가 서버(20)에 위치하고 있는 오브젝트(Object)의 참조 포인터를 얻은 후 그 오브젝트(Object)의 매소드(Method)를 호출하게 된다.

<20> 이때 데이터베이스 접근이라는 특수성으로 인해 전달되는 파라미터는 SQL 문장이 되며, 반환되는 데이터 구조는 IDL에서 정의한 구조체 포인터를 Call by reference로 전달된다.

<21> 전달된 SQL 데이터를 추출하며, 그 데이터는 파라미터로 전달받은 구조체 레퍼런스(Reference)의 시퀀스(Sequence) 형으로 메모리를 할당한 후 데이터를 복사하게 된다.

<22> 해당 매소드(Method)에 대한 처리가 종료되면, 클라이언트(10)는 파라미터로 전달한 구조체 레퍼런스(Reference)를 통해 반환값을 가져와 처리할 수 있게 된다.

<23> 클라이언트(10)가 다른 트랜잭션(Transaction)(SELECT SQL)을 요구하기 위해서는 IDL에 정의되어 있는 해당되는 특정 매소드(Method)를 호출하며, 반환되는 데이터의 구조도 그 트랜잭션 매소드(Transaction method)를 위해 모두 IDL에 정의되어야 한다.

<24> 그래서 종래의 CORBA 플랫폼 하에서의 서버(20)와 클라이언트(10) 구조에서는 클라이언트(10)가 서버(20)에 동작하도록 요구하는 데이터베이스 처리 오브젝트(Object)의 매소드(Method)를 모두 IDL의 인터페이스에 명시한 후 그 범위 내에서 코드를 구현하게 된다.

<25> 그러나 종래의 기술은 새롭게 정의되는 검색 조건에 대해서는 추가적인 IDL을 정의한 후 재 컴파일을 통해 구현하여야만 하는 불편함이 있었다.

<26> 특히 각각의 선택(SELECT) 요구시 해당 매소드(Method)의 IDL에는 클라이언트가 수신하고자 하는 데이터 구조가 명시적으로 기술되어야 하며, 이러한 데이터 구조는 요구 시 계속적으로 추가되어 IDL 및 추출되는 오브젝트(Object)의 크기도 전체적으로 커지는 문제점이 있었다.

<27> 이는 선택 트랜잭션(SELECT Transaction)의 특성상 반환되는 데이터의 구조 및 데이터베이스 Query를 Server에서 동적으로 처리하는 것에 어려움이 있기 때문인데, 종래에는 이러한 문제점을 해결하지 못하였다.

【발명이 이루고자 하는 기술적 과제】

<28> 이에 본 발명은 상기와 같은 종래의 제반 문제점을 해소하기 위해 제안된 것으로, 본 발명의 목적은 CORBA를 이용한 분산 처리 시스템에서 서버와 클라이언트 간에 데이터 처리에서 데이터베이스 검색을 위한 범용 IDL을 정의하고 서버에서의 처리를 동적으로 처리할 수 있게 함으로써 개발자의 개발 범위를 최소화하고 이후에 변경되는 클라이언트의 트랜잭션 요구에 대해서 IDL의 변경을 최소화할 수 있는 CORBA 플랫폼을 이용한 동적 데이터베이스 처리방법을 제공하는데 있다.

<29> 상기와 같은 목적을 달성하기 위하여 본 발명의 일 실시예에 의한 CORBA 플랫폼을 이용한 동적 데이터베이스 처리방법은,

<30> 클라이언트에서는 자신이 데이터베이스에 접근하기 위해 SQL 스트링을 구성한 다음 구성한 스트링과 참조 포인터를 파라미터로 서버의 범용 매소드를 호출하는 제 1 단계와; 상기 서버의 매소드는 SQL 스트링을 이용하여 데이터를 질의하여 데이터베이스로부터 받아온 하나의 row 데이터를 로컬 메모리에 링크드 리스트 형태로 저장하고, 한의 row에 대한 선택 작업 완료 후 범용 데이터 형의 클래스를 인스턴스 생성하고, 어레이로 메모리를 생성시켜 링크드 리스트에 있는 데이터를 할당하는 제 2 단계와; 상기 제 2 단계에서의 질의 작업이 완료되면 상

기 클라이언트는 참조 포인터를 이용해 상기 서버가 할당한 결과값에 접근하는 제 3 단계를 포함하여 수행함을 그 기술적 구성상의 특징으로 한다.

【발명의 구성 및 작용】

<31> 이하, 상기와 같은 본 발명, CORBA 플랫폼을 이용한 동적 데이터베이스 처리방법의 기술적 사상에 따른 일 실시예를 도면을 참조하여 설명하면 다음과 같다.

<32> 도 4는 본 발명에 의한 CORBA 플랫폼을 이용한 동적 데이터베이스 처리방법을 보인 흐름도 이다.

<33> 이에 도시된 바와 같이, 클라이언트(10)에서는 자신이 데이터베이스에 접근하기 위해 SQL 스트링을 구성한 다음 구성한 스트링과 참조 포인터를 파라미터로 서버(20)의 범용 매소드를 호출하는 제 1 단계(ST21)(ST22)와; 상기 서버(20)의 매소드는 SQL 스트링을 이용하여 데이터를 질의하여 데이터베이스로부터 받아온 하나의 row 데이터를 로컬 메모리에 링크드 리스트 형태로 저장하고, 한의 row에 대한 선택 작업 완료 후 범용 데이터 형의 클래스를 인스턴스 생성하고, 어레이로 메모리를 생성시켜 링크드 리스트에 있는 데이터를 할당하는 제 2 단계(ST23 ~ ST26)와; 상기 제 2 단계에서의 질의 작업이 완료되면 상기 클라이언트(10)는 참조 포인터를 이용해 상기 서버(20)가 할당한 결과값에 접근하는 제 3 단계 (ST27)(ST28)를 포함하여 수행한다.

<34> 이와 같이 구성된 본 발명에 의한 CORBA 플랫폼을 이용한 동적 데이터베이스 처리방법의 동작을 첨부한 도면에 의거 상세히 설명하면 다음과 같다.

<35> 먼저 CORBA를 이용한 분산 처리 시스템에서 서버(20)와 클라이언트(20) 간에 데이터 처리 구조는 서버(20)에 있는 오브젝트(Object)의 매소드(Method)를 원격의 클라이언트(Client)(10)가 호출하게 되면, 실제 동작은 서버(20)의 오브젝트(Object)에서 일어나고, 그 결과 데이터를 IIOP(Internet Inter-ORB Protocol)를 통해 전송하게 된다.

<36> 이러한 구조에서 서버(20)에서 동작하는 데이터베이스 트랜잭션 오브젝트(Transaction Object)에 클라이언트(10)가 접근하여 결과 데이터를 반환 받기 위해서 각각의 트랜잭션(Transaction) 종류별로 IDL을 모두 정의해야 하는 문제점이 있다. 이런 경우 클라이언트(10)에서 데이터베이스를 검색하여 수신하는 데이터의 구조가 변경되는 경우에 IDL을 다시 정의해야 한다. 이런 문제점을 해결하기 위해 클라이언트(10)와 서버(20) 사이에 데이터베이스 검색을 위한 범용 IDL을 정의하고 서버(20)에서의 처리를 동적으로 처리하는 메커니즘을 제시하여 개발자의 개발 범위를 최소화하고 이후에 변경되는 클라이언트(10)의 트랜잭션 요구에 대해서 IDL의 변경을 최소화하고자 한다.

<37> 그래서 본 발명은 종래의 기술이 갖는 CORBA 서버(20)에서의 데이터베이스 SELECT Query 처리 시 발생하는 데이터의 동적 바인딩(dynamic binding) 현상(데이터를 추출할 때까지는 형이나 구조를 알 수 없는 문제)으로 인해 범용적인 IDL 구현에 어려움이 따르기 때문에 트랜잭션(Transaction)을 처리하는 서버(20)의 SELECT Query Method를 동적으로 처리하는 범용 Method로 구성하고 그에 따른 IDL을 범용으로 선언한다.

<38> 그리고 본 발명은 범용 IDL 및 IDL 컴파일 시 추출되는 스터브(Stub) 및 스케레톤 코드(Skeleton code) 그리고 동적 SQL을 처리하는 서버(20) 쪽의 매소드(Method)로 이루어져 있다.

<39> 동적 SQL을 처리하는 매소드(Method)는 검색 대상이 되는 테이블의 데이터를 읽어 오는데 정적인 변수를 사용하는 대신 데이터베이스에서 제공하는 동적(Dynamic) SQL 기술을 사용하여 데이터베이스의 동적인 임시 메모리에 읽어온 질의 데이터를 저장하여 테이블의 구조와는 관계없이 데이터를 저장한 후 반환하기 위하여 CORBA의 Any type의 변수에 할당한다.

<40> 이 Any type의 변수는 다시 하나의 row를 구성하기 위해 시퀀스 타입(sequence type)으로 구조화되며, 다수의 row를 반환하기 위해 시퀀스 타입(sequence type)을 한번 더 IDL에서 정의한다.

<41> 동적 메모리는 테이블의 구조(테이블의 컬럼 구조)에 따라 자동적으로 할당되고 소멸된다.

<42> 그리고 도 5는 도 4에 의한 CORBA 플랫폼에서의 동적 처리 구조를 보인 도면이고, 도 6은 도 4에 의한 동적 서버의 동작 구조를 보인 도면이며, 도 7은 도 4에 의한 범용 매소드의 IDL과 리턴값의 구조를 보인 도면이다.

<43> 그래서 범용 IDL에서 정의한 오브젝트(Object)의 매소드(Method)를 클라이언트(10)가 원하는 SQL 스트링(string)의 파라미터로 전달한다.

<44> 이 파라미터를 받은 서버(20) 쪽의 오브젝트(Object)는 이 SQL 문장을 동적(Dynamic) SQL 모드로 구동시켜 해당 데이터를 추출한다.

<45> 추출된 데이터는 한 행씩 동적으로 생성되는 메모리에 적재되며, 모든 트랜잭션(Transaction)이 종료되면 동적 메모리에 저장된 데이터를 반환하기 위해 CORBA에서 제공하는 시퀀스(sequence) 형태의 any type 변수에 할당한다.

<46> 매소드(Method)가 종료되면, 클라이언트(10)의 호출한 매소드(Method)는 call by reference로 반환 파라미터를 넘겼기 때문에 시퀀스(sequence) 형태의 원하는 SQL 결과값을 IIOP를 통해 수신한다.

<47> 이러한 본 발명의 동작을 순서대로 설명하면 다음과 같다.

<48> 1. 클라이언트(10)에서는 자신이 데이터베이스에 접근하기 위해 SQL string을 구성한다.

<49> 2. 클라이언트(10)는 구성한 SQL string 과 그 결과값을 반환 받기 위해 IDL에서 정의한 범용 데이터 형의 참조 포인터를 파라미터로 서버(20)의 범용 Method를 호출한다.

<50> 3. 파라미터를 받은 서버(20)의 Method는 SQL string을 이용하여 데이터를 질의(Query) 한다.

<51> 4. 데이터베이스로부터 받아온 하나의 row 데이터를 로컬 메모리(local memory)에 링크드 리스트(Linked list) 형태로 저장한다.

<52> 5. 하나의 row에 대한 Select 작업 완료 후 범용 데이터 형의 클래스(Class)를 인스턴스 생성(Instantiation) 한다.

<53> 6. 시퀀스 타입(sequence type)이므로 마치 어레이(array) 처럼 메모리를 생성시켜 linked list에 있는 데이터를 할당한다.

<54> 7. 3번부터 6번까지 작업은 Query 작업이 끝날 때까지 반복 수행한다.

<55> 8. 매소드(Method)가 완료되면 클라이언트(10)는 참조 포인터(reference pointer)를 이용해 서버(20)가 할당한 결과값에 접근이 가능하다.

<56> 이처럼 본 발명은 CORBA를 이용한 분산 처리 시스템에서 서버와 클라이언트 간에 데이터 처리에서 데이터베이스 검색을 위한 범용 IDL을 정의하고 서버에서의 처리를 동적으로 처리할 수 있게 함으로써 개발자의 개발 범위를 최소화하고 이후에 변경되는 클라이언트의 트랜잭션 요구에 대해서 IDL의 변경을 최소화하게 되는 것이다.

<57> 이상에서 본 발명의 바람직한 실시예를 설명하였으나, 본 발명은 다양한 변화와 변경 및 균등물을 사용할 수 있다. 본 발명은 상기 실시예를 적절히 변형하여 동일하게 응용할 수 있음이 명확하다. 따라서 상기 기재 내용은 하기 특허청구범위의 한계에 의해 정해지는 본 발명의 범위를 한정하는 것이 아니다.

【발명의 효과】

<58> 이상에서 살펴본 바와 같이, 본 발명에 의한 CORBA 플랫폼을 이용한 동적 데이터베이스 처리방법은 CORBA를 이용한 분산 처리 시스템에서 서버와 클라이언트 간에 데이터 처리에서 데이터베이스 검색을 위한 범용 IDL을 정의하고 서버에서의 처리를 동적으로 처리할 수 있게 함으로써 개발자의 개발 범위를 최소화하고 이후에 변경되는 클라이언트의 트랜잭션 요구에 대해서 IDL의 변경을 최소화할 수 있는 효과가 있게 된다.

<59> 또한 본 발명은 클라이언트가 어떤 SQL을 수행하든지 하나의 범용 매소드만으로도 처리가 가능하며, 하나의 IDL만 정의하면 되기 때문에 IDL 정의에 드는 개발 기간 및 비용이 줄어들게 되고, CORBA를 이용한 분산 트랜잭션 처리가 가능하여 프로세스의 성능이 향상되는 효과도 있게 된다.

【특허청구범위】

【청구항 1】

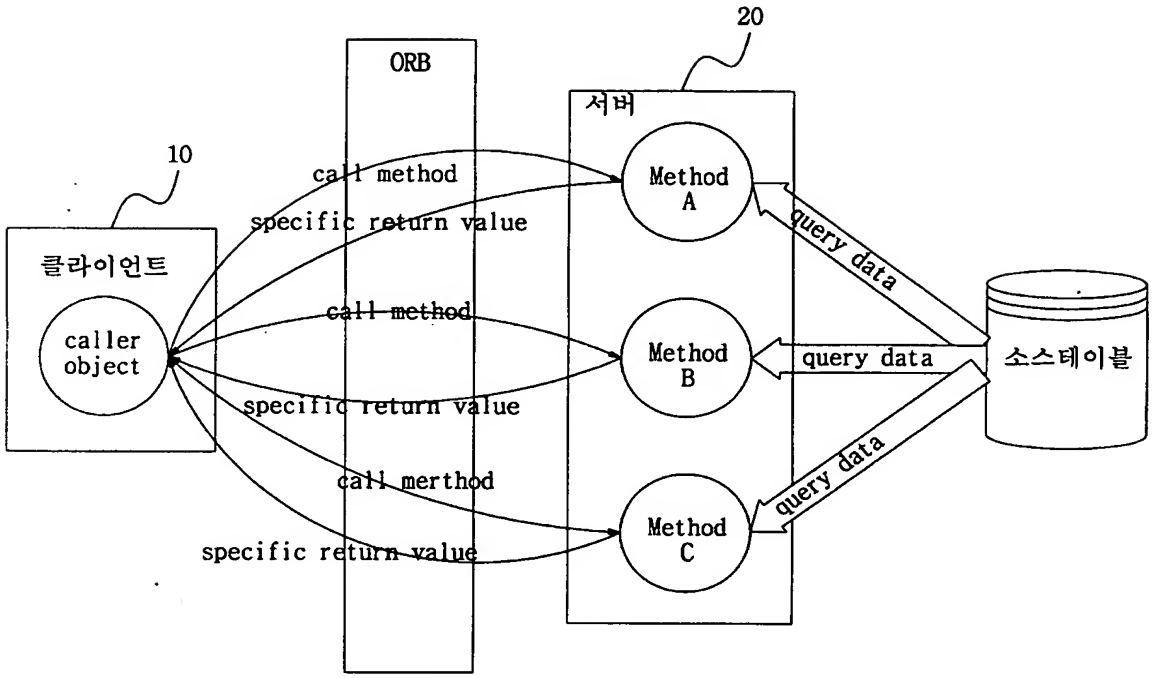
클라이언트에서는 자신이 데이터베이스에 접근하기 위해 SQL 스트링을 구성한 다음 구성한 스트링과 참조 포인터를 파라미터로 서버의 범용 매소드를 호출하는 제 1 단계와;

상기 서버의 매소드는 SQL 스트링을 이용하여 데이터를 질의하여 데이터베이스로부터 받아온 하나의 row 데이터를 로컬 메모리에 링크드 리스트 형태로 저장하고, 한의 row에 대한 선택 작업 완료 후 범용 데이터 형의 클래스를 인스턴스 생성하고, 어레이로 메모리를 생성시켜 링크드 리스트에 있는 데이터를 할당하는 제 2 단계와;

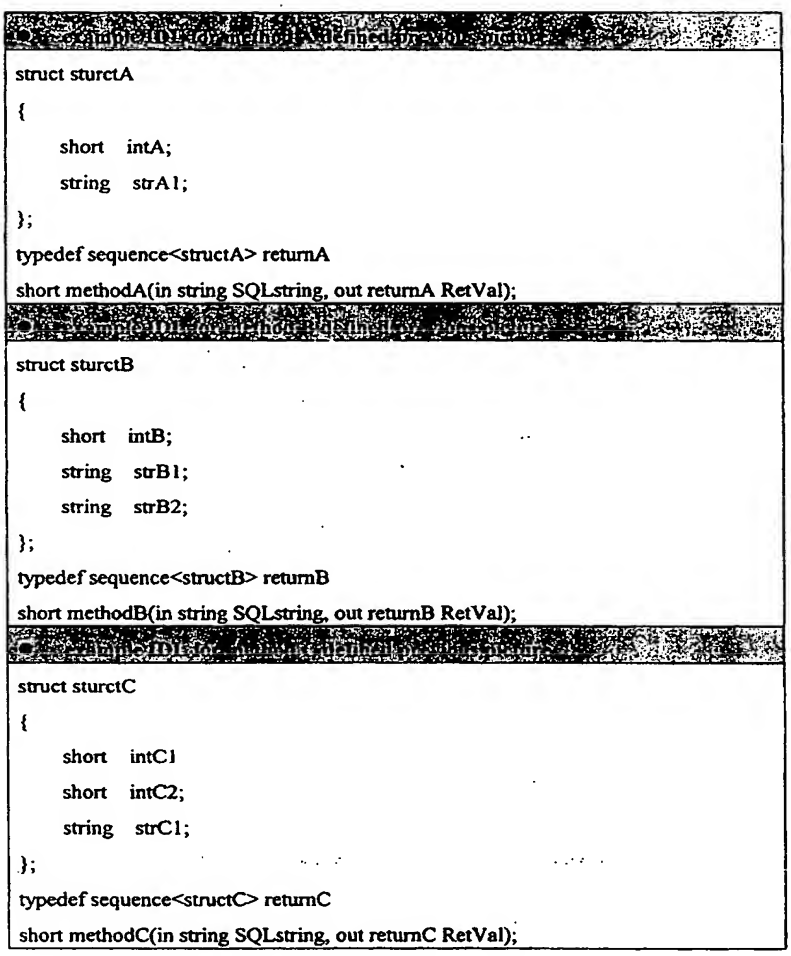
상기 제 2 단계에서의 질의 작업이 완료되면 상기 클라이언트는 참조 포인터를 이용해 상기 서버가 할당한 결과값에 접근하는 제 3 단계를 포함하여 수행하는 것을 특징으로 하는 코바 플랫폼을 이용한 동적 데이터베이스 처리방법.

【도면】

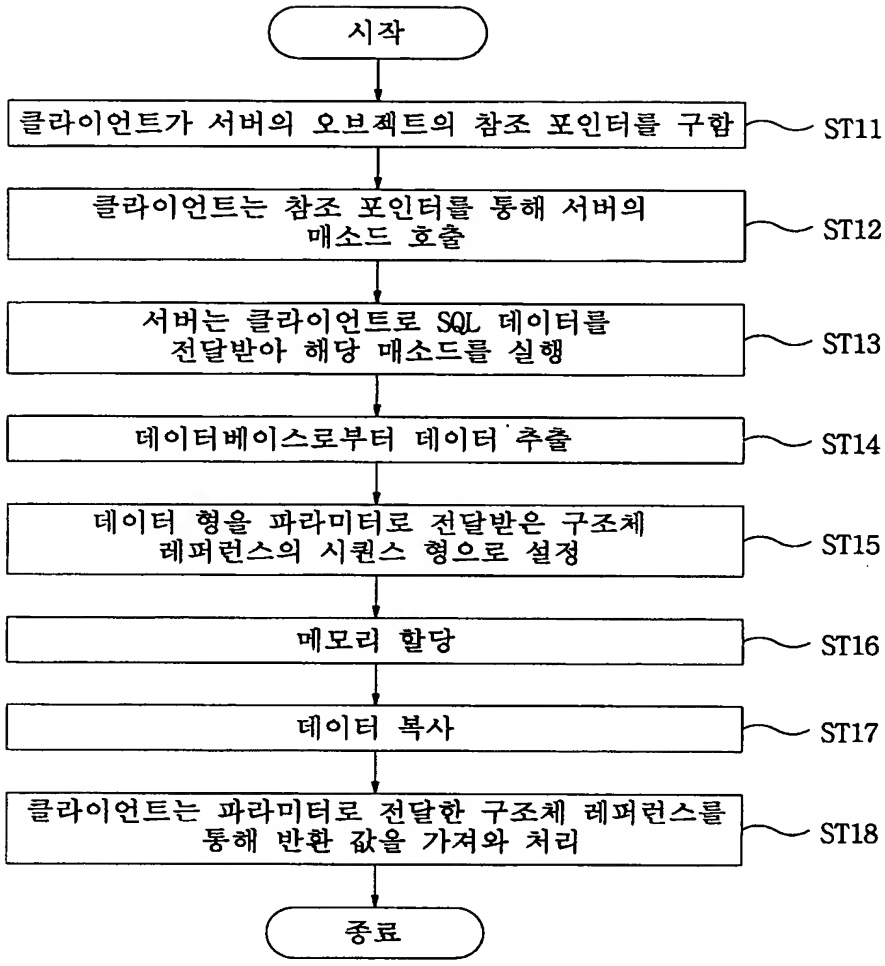
【도 1】


20
ORB
서버
10
call method
specific return value
Method A
query data
클라이언트
caller object
call method
Method B
query data
소스테이블
specific return value
call merthod
query data
Method C
specific return value

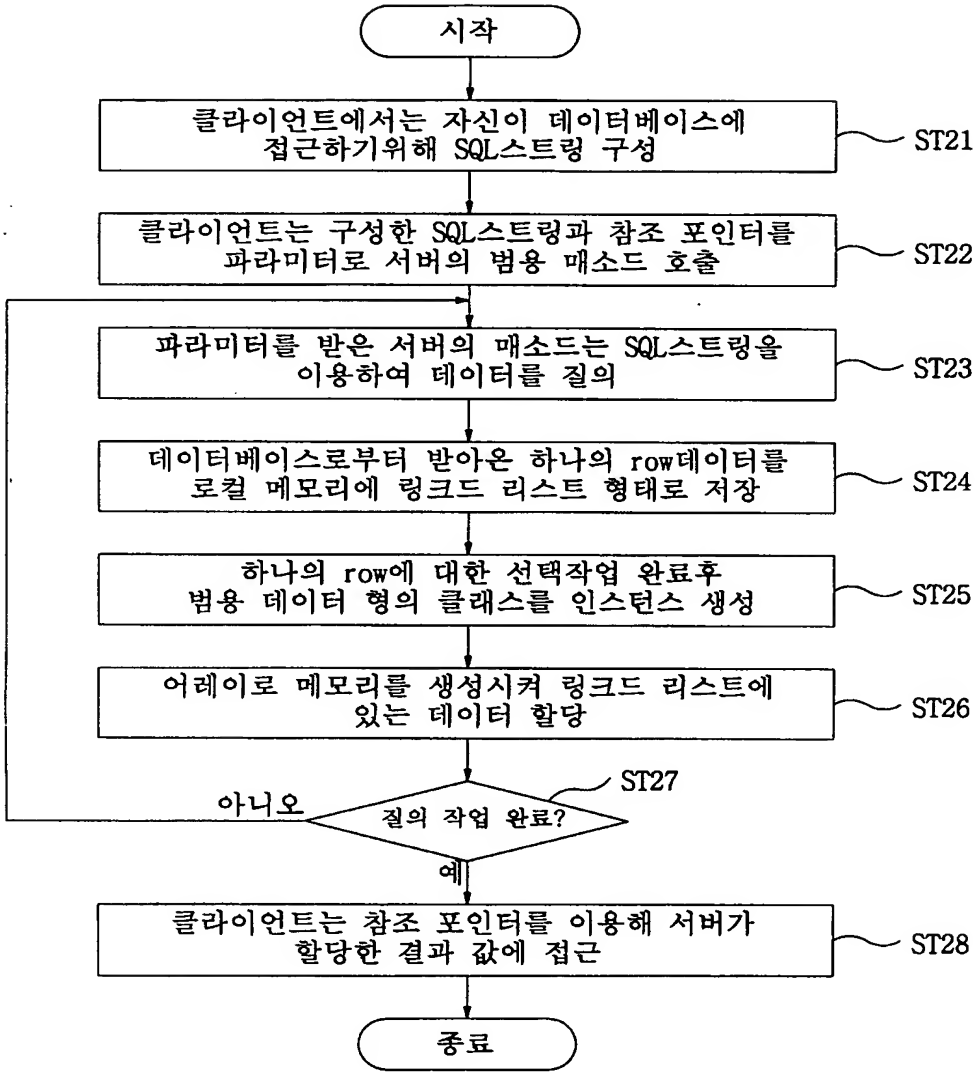
【도 2】


struct sturctA
{
short intA;
string strA1;
};
typedef sequence<structA> returnA
short methodA(in string SQLstring, out returnA RetVal);
struct sturctB
{
short intB;
string strB1;
string strB2;
};
typedef sequence<structB> returnB
short methodB(in string SQLstring, out returnB RetVal);
struct sturctC
{
short intC1
short intC2;
string strC1;
};
typedef sequence<structC> returnC
short methodC(in string SQLstring, out returnC RetVal);

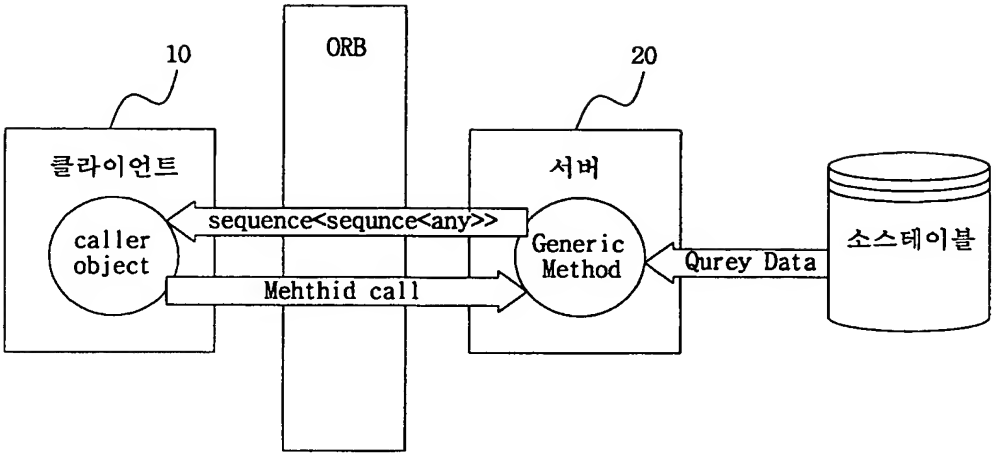
【도 3】


시작
클라이언트가 서버의 오브젝트의 참조 포인터를 구함
ST11
클라이언트는 참조 포인터를 통해 서버의 매소드 호출
ST12
서버는 클라이언트로 SQL 데이터를 전달받아 해당 매소드를 실행
ST13
데이터베이스로부터 데이터 추출
ST14
데이터 형을 파라미터로 전달받은 구조체 레퍼런스의 시퀀스 형으로 설정
ST15
메모리 할당
ST16
데이터 복사
ST17
클라이언트는 파라미터로 전달한 구조체 레퍼런스를 통해 반환 값을 가져와 처리
ST18
종료

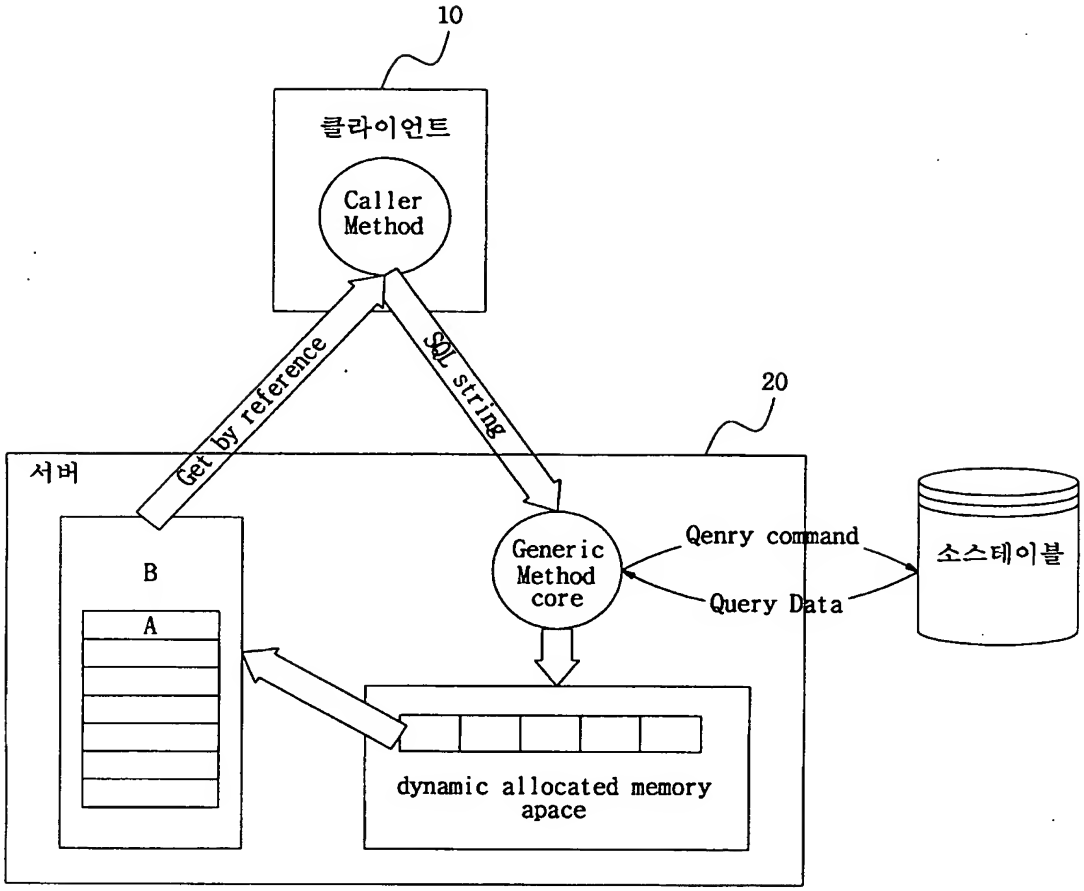
【도 4】


시작
클라이언트에서는 자신이 데이터베이스에 접근하기위해 SQL스트링 구성
ST21
클라이언트는 구성한 SQL스트링과 참조 포인터를 파라미터로 서버의 범용 매소드 호출
ST22
파라미터를 받은 서버의 매소드는 SQL스트링을 이용하여 데이터를 질의
ST23
데이터베이스로부터 받아온 하나의 row데이터를 로컬 메모리에 링크드 리스트 형태로 저장
ST24
하나의 row에 대한 선택작업 완료후 범용 데이터 형의 클래스를 인스턴스 생성
ST25
어레이로 메모리를 생성시켜 링크드 리스트에 있는 데이터 할당
ST26
ST27
아니오
질의 작업 완료?
예
클라이언트는 참조 포인터를 이용해 서버가 할당한 결과 값에 접근
ST28
종료

【도 5】


10
ORB
20
클라이언트
서버
caller object
sequence<sequnce<any>>
Mehthid call
Generic Method
Qurey Data
소스테이블

【도 6】



【도 7】

example IDL for generic method defined previous picture

```
struct elementValue
{
    short   type;          // 1 : integer   2 : float   3 : string   4 : decimal   5 : date
    any   value;           // any type for real value
};
typedef sequence<structA> ElemSeq          // structure of A
typedef sequence<ElemSeq> ReturnSeq        // structure of B
short genericMethod(in string SQLstring, out ReturnSeq RetVal);
```